A Context-Aware Medical Query System (MediBot): Bridging Large Language Models and Clinically Validated Documentation

[1]Satya Prakash, [2]Kundan Kumar Jaiswal, [3]Shivang Saxena,[4]Gourav Thoke, [5]Mohit Sahu,[6]Mohit Meena, [7]Shiv Purwar

[1,2,3,4,5,6,7]B. Tech Scholar, *Department of CSE-AIML, OIST Bhopal, 462021*
Corresponding Author Email- kundanjai399kumar@gmail.com

ABSTRACT:

The rapid advancement of Large Language Models (LLMs) has revolutionized the field of natural language processing, offering unprecedented capabilities in text generation and human-like interaction. However, in the high-stakes domain of healthcare, the deployment of generic LLMs is severely hindered by the "hallucination" phenomenon the tendency of models to generate factually incorrect or commercially biased information with high confidence **[1]**. This research introduces MediBot, a specialized medical chatbot architecture that utilizes Retrieval-Augmented Generation (RAG) to ground AI responses in clinically validated literature.

The primary motivation behind MediBot is to address the data transparency gap. While models like GPT-4 or Llama 3 are trained on vast internet datasets, they often lack access to the most recent clinical guidelines or proprietary medical texts **[2]**. MediBot solves this by decoupling the knowledge base from the reasoning engine. By implementing a modular three-phase pipeline Knowledge Extraction, Vector Space Memory Setup, and Retrieval-QA Integration MediBot ensures that every answer provided to a user is derived from a specific, traceable source document.

Keywords: Medical Chatbot, Retrieval Augmented Generation (RAG), Groq LPU, FAISS, LangChain, Evidence-Based Medicine.

INTRODUCTION:

The landscape of modern healthcare is currently undergoing a paradigm shift, transitioning from traditional reactive models to the data-driven, personalized era known as HealthCare 5.0 **[4]**. At the heart of this transformation is the integration of Artificial Intelligence (AI) and Large Language Models (LLMs), which promise to alleviate the immense pressure on global medical systems. However, as we integrate these black-box models into clinical settings, we face a fundamental paradox: the very models that offer the most human-like interaction are often the least reliable when it comes to factual precision. This research introduces MediBot, a Retrieval-Augmented Generation (RAG) framework designed to reconcile the conversational power of LLMs with the non-negotiable requirement for clinical accuracy.

1. The Crisis of Information Overload and the LLM Gap: -

For decades, the primary challenge in medicine was the accessibility of information. Today, the challenge has flipped; it is now the sheer volume of information that overwhelms practitioners. It is estimated that the volume of medical knowledge doubles every 73 days, making it humanly impossible for clinicians to stay updated on every new study, guideline, or drug interaction **[9]**.

Early attempts to solve this involved "expert systems" rule-based programs that followed "if-then" logic. While accurate, they were brittle and could not understand the nuance of natural language. The arrival of LLMs like GPT-4

and Llama 3 changed everything by providing a fluid interface that could synthesize information. Yet, these models suffer from "hallucinations" a phenomenon where the model generates plausible-sounding but entirely fabricated medical data **[1]**. In a field where an incorrect dosage recommendation can be fatal, "plausible" is not good enough.

2. The Socioeconomic Necessity for Evidence-Based Chatbots: -

The need for MediBot is not merely technical; it is a matter of global health equity. In resource-limited settings, such as rural Bangladesh or parts of Sub-Saharan Africa, the ratio of physicians to patients is alarmingly low **[5]**. In these regions, adolescents and marginalized groups often turn to the internet for sensitive information regarding reproductive health, mental health, or chronic disease management.

Generic AI often provides Western-centric or outdated advice. By using the RAG architecture implemented in MediBot, we can load local clinical guidelines and peer-reviewed regional studies into the "memory" of the bot. This ensures that the advice given is not only scientifically sound but also contextually relevant to the specific demographic being served.

3. Defining Retrieval-Augmented Generation (RAG): -

To understand why MediBot is superior to standard chatbots, we must define the RAG mechanism. Traditional LLMs rely on "parametric memory" knowledge frozen in time during the model's training phase. RAG introduces "non-parametric memory," which allows the model to look up information in real-time from an external, curated database **[10]**.

This process involves converting unstructured PDF data into mathematical vectors. When a user asks a question, the system does not ask the LLM for the answer immediately. Instead, it performs a semantic search in the vector database to find the most relevant "evidence" and presents that evidence to the LLM as a context window. The LLM then acts as a sophisticated translator, turning that raw evidence into a natural, easy-to-understand response.

4. The Technical Blueprint: MediBot's Three-Phase Architecture: -

This research paper details the implementation of MediBot across three distinct phases, reflecting a modular approach to AI development.

Phase 1: Knowledge Ingestion and Vectorization. Using FAISS (Facebook AI Similarity Search) and the all-MiniLM-L6-v2 embedding model, MediBot creates a high-dimensional map of medical knowledge. The use of a 500-character chunk size with a 50-character overlap is a strategic choice made to ensure that no sentence is cut off mid-thought, preserving the clinical context **[3]**.

Phase 2: Semantic Bridge and Retrieval. Through the Groq Inference engine and Llama 3.1, the system achieves sub-second latency. This is crucial for clinical environments where time is of the essence.

Phase 3: Human-Centric Interface. The deployment via Streamlit allows for a "traceable" UI. By providing the user with the source documents used to generate the answer, MediBot restores the trust that is often lost in AI interactions **[6]**.

5. Comparison of Methodologies: -

To justify the choice of RAG over other methods like "Fine-Tuning," it is helpful to look at the following comparison table which highlights why MediBot's architecture was chosen for this research:

Table 1: Comparison of AI Adaptation Strategies for Medical Use

Feature	Generic LLM (Base)	Fine-Tuned LLM	RAG (MediBot)
Knowledge Update	Requires retraining	Requires expensive compute	Near-instant (update PDF)
Accuracy	Low (Hallucinations)	Moderate	High (Source-grounded)
Traceability	None	Limited	Full (Direct Citations)
Implementation Cost	Low	High	Moderate
Data Privacy	Public Cloud Risks	High	High (Can be Local)

Performance Evaluation: -

To illustrate the trade-offs and benefits of MediBot's design, consider the following conceptual latency comparison between typical GPU-based RAG systems and the Groq LPU-accelerated pipeline:

Table 2: Inference Latency Benchmarks

System	Retrieval Time (ms)	Generation Time (ms)	Total Latency (ms)	Accuracy (%)
GPU RAG Baseline	200	800	1000	87
Optimized RAG with GPU	150	600	750	89
MediBot (Groq LPU)	150	**200**	**350**	89

6. Problem Statement and Research Objectives: -

Despite the promise of RAG, challenges remain regarding the "retrieval quality." If the system retrieves irrelevant documents, the LLM will generate an irrelevant answer a problem known as "Garbage In, Garbage Out." Therefore, the primary objective of this research is to evaluate the effectiveness of the MediBot framework in:

Reducing factual errors in medical Q&A.

Providing a low-latency interface suitable for real-time clinical support.

Establishing a "source-first" culture in medical AI where every claim is backed by a document.

7. Conclusion of Introduction: -

By moving away from a single, monolithic intelligence toward a modular, evidence-based retrieval system, we can create tools that are not just "smart," but "wise." MediBot represents a significant step toward this goal, offering a scalable solution that can be adapted for pharmacy education **[6]**, specialized surgery guidelines **[2]**, or adolescent mental health support **[5]**. The following sections of this paper will detail the experimental setup, the code-level implementation, and a rigorous evaluation of the system's performance metrics.

LITERATURE REVIEW:

Ahamed, S., et al. (2026) presented *SuSastho.AI*, emphasizing that standard LLMs struggle with medical accuracy in specific cultural contexts. By implementing RAG with a clinically validated knowledge base, they achieved an accuracy rate of 86.7%, proving that external knowledge retrieval is essential for reducing incorrect responses.

Nayinzira, J. P., & Adda, M. (2024) explored the architecture of mental health chatbots, demonstrating that advanced retrieval strategies (like Multi-query RAG) significantly outperform naive approaches. Their work highlights the necessity of semantic search over simple keyword matching for understanding patient intent.

Shamim Ahamed., et al. (2026) SuSastho.AI uses LLMs to provide culturally sensitive, evidence-based health support for Bengali adolescents. Pilot results show improved health literacy, with scope to enhance language accuracy. The study highlights AI's potential to improve access to reliable health information in low-resource settings.

Younesi, A., et al. (2026) surveyed *HealthCare 5.0*, envisioning a future where medical systems move from reactive care to proactive "Sense-Transmit-Reason-Act" loops. In this context, MedLLMs serve as the "reasoning" engine. Our project aligns with this vision by using Groq's high-speed inference to minimize the "Transmit" and "Reason" latency.

Steybe, D., et al. (2025) evaluated *GuideGPT*, a chatbot for diagnosing jaw osteonecrosis. Their study confirmed that RAG-based systems scored significantly higher in scientific explanation ($p=0.032$) compared to generic GPT-4 models, validating the RAG approach for specialized clinical domains.

Abolfazl Younesi., et al. (2026) The paper outlines a roadmap for HealthCare 5.0 by integrating IoT, AI, and 6G to enable smart monitoring, predictive care, autonomous diagnostics, and personalized treatment. It stresses the importance of collaboration, data privacy, and explainable AI, and highlights key research areas needed for sustainable, patient-centric healthcare systems.

S. Bakhaya., et al. (2026) This study highlights the effectiveness of AI chatbots in improving pharmacy students' written communication skills for self-care consultations. It also acknowledges the use of ChatGPT-4o for language and reference enhancement, with authors retaining full responsibility for the final content.

METHODOLOGY:

The proposed MediBot system follows a standard RAG architecture enhanced for speed and local privacy. The methodology is divided into three core phases: Knowledge Ingestion, Vector Space Construction, and the Retrieval-QA Orchestration. By decoupling the static knowledge (PDFs) from the reasoning engine (LLM), the system ensures that clinical responses are grounded in verifiable evidence.

1. Architectural Framework

The system follows a modular architecture designed for scalability and low-latency interaction. The pipeline is built using the LangChain framework, which coordinates the data flow between the document loaders, the vector database, and the generative model.

Table 1: Technical Stack Specifications: -

Component	Choice	Practical Role in the System
Orchestration	**LangChain**	Acts as the "glue." It manages the logic flow: taking user input, retrieving relevant documents from FAISS, and passing that context to Groq.
Embedding Model	**all-MiniLM-L6-v2**	Converts text into numerical vectors (384 dimensions). It is specifically chosen because it is lightweight and fast, making it ideal for real-time applications without requiring massive GPU resources.
Vector Database	**FAISS (CPU)**	Stores your clinical documents as vectors. Using the CPU version suggests the database size is likely manageable within RAM and avoids the overhead/cost of dedicated GPU clusters while maintaining microsecond search speeds.
Inference Engine	**Groq (Llama 3.1 8B)**	This is the "brain." Groq's LPU (Language Processing Unit) architecture allows Llama 3.1 to generate text at speeds often exceeding 500 tokens per second, which is essential for "sub-second" clinical decision support.
User Interface	**Streamlit**	Provides the front-end. It allows doctors or researchers to interact with the model via a web browser, supporting features like file uploads (PDFs) and chat interfaces with minimal backend code.

2. Phase I: Knowledge Extraction and Pre-processing

The accuracy of a RAG system is fundamentally limited by the quality of its retrieval corpus. MediBot utilizes a "Directory Loading" strategy to ingest unstructured medical data.

2.1 Document Loading

Using DirectoryLoader combined with PyPDFLoader, the system recursively scans a designated data directory for medical textbooks and clinical guidelines. This allows the system to remain "domain-agnostic," meaning it can be updated simply by swapping the PDF files without retraining the model **[3]**.

2.2 Recursive Character Text Splitting

Medical literature often contains complex, multi-clause sentences. To maintain semantic integrity, we implemented a **Recursive Character Text Splitter**.

Chunk Size: 500 characters.

Chunk Overlap: 50 characters. The 10% overlap is a critical design choice; it ensures that context spanning across chunk boundaries such as a drug name followed by its contraindications is preserved in both adjacent vectors, reducing the risk of "context fragmentation" **[7]**.

3. Phase II: Vector Space and Embedding Generation

Once the text is segmented, it must be converted into a mathematical format that the computer can understand.

3.1 Sentence Embeddings

We utilize the sentence-transformers/all-MiniLM-L6-v2 model. This model maps sentences and paragraphs to a 384-dimensional dense vector space. Unlike keyword-based search (which looks for exact words), this embedding-based approach enables Semantic Search. For example, if a user asks about "high blood pressure," the system can retrieve documents discussing "hypertension" because their vectors are mathematically close in hyperspace **[10]**.

3.2 FAISS Indexing

The generated vectors are stored in a FAISS (Facebook AI Similarity Search) index. FAISS uses L2 (Euclidean) distance to calculate similarity. The index is saved locally as a index file, allowing the bot to boot up in seconds by loading the pre-calculated "memory" rather than re-processing the PDFs every time **[2]**.

4. Phase III: The Retrieval-QA Chain

The final phase involves the interactive loop where user queries are transformed into evidence-based answers.

4.1 The RAG Logic Loop

When a user submits a prompt via the Streamlit interface:

Query Vectorization: The user's question is converted into a vector using the same MiniLM model.

Top-K Retrieval: The system performs a similarity search against the FAISS index to retrieve the **Top-3 (k=3)** most relevant document chunks.

Prompt Augmentation: These chunks are injected into a specialized prompt template:

"Use the following pieces of context to answer the question at the end. If you don't know the answer, just say you don't know, don't try to make up an answer."

Generative Inference: The augmented prompt is sent to the **Llama-3.1-8b-instant** model via the **Groq Cloud API**.

4.2 Groq-Inference Acceleration

To solve the latency issues common in medical chatbots, we integrated the Groq LPU (Language Processing Unit). This allows the system to achieve token generation speeds far exceeding standard GPU-based inference, making the bot feel

"instantaneous" and more human-like in conversation [4].

5. Visual Representation of Methodology

Flowchart 1: The Offline Ingestion Pipeline (create_memory_for_llm.py)

Flowchart 2: The Online Inference Pipeline (medibot.py)

6. Summary of Parameters

For the sake of reproducibility, the following hyper-parameters were kept constant during testing:

Table 2: System Hyper-parameters :

Parameter	Meaning & Impact
Temperature (0.5)	**The "Creativity" Dial.** At 0.5, the model is in a "balanced" mode. It isn't strictly robotic (0.0), but it isn't hallucinating wild stories (1.0). It is ideal for customer support or general-purpose assistants where you want natural flow without sacrificing facts.
Max Tokens (512)	**The "Breath" Limit.** This limits the response length to roughly 350–400 words. It's enough for a detailed multi-paragraph explanation, but it will cut off if the answer requires a very long essay.
Search Type (Similarity)	**The "Meaning" Matcher.** Instead of looking for exact keywords (like "blue car"), it looks for mathematical proximity in meaning. It can find a "navy vehicle" even if the word "blue" isn't in the document.
K-Value (3)	**The "Context" Breadth.** The AI will look at its database and pull the **top 3** most relevant snippets to help answer your question. This is a "lean" setting that keeps the prompt clean and prevents the AI from getting confused by too much data.
Normalization (L2)	**The "Fairness" Scalar.** L2 normalization (Euclidean) scales all data vectors to a length of 1. This ensures that a very long document doesn't get ranked higher just because it has more words; it focuses strictly on the *direction* of the meaning.

IMPLEMENTATION DETAILS:

The implementation of MediBot follows a structured three-phase pipeline, designed to transform unstructured medical literature into a queryable knowledge base. The following details describe the technical configurations, software environments, and procedural steps used in the development of the system.

1. Development Environment and Software Stack

The project is built using Python 3.12 and managed with a project-oriented architecture. The following key libraries were used:

LangChain: Utilized as the primary framework for coordinating the RAG pipeline.

FAISS (CPU): A high-performance library for efficient similarity search and clustering of dense vectors.

Streamlit: Employed to build the reactive web interface for real-time user interaction.

HuggingFace Embeddings: Specifically the sentence-transformers/all-MiniLM-L6-v2 model for generating 384-dimensional dense vectors.

Groq Inference Engine: Used to host the llama-3.1-8b-instant model, ensuring low-latency response generation.

2. Phase 1: Knowledge Ingestion and Memory Creation

The first stage involves preparing the medical knowledge base for retrieval. This is handled by the create_memory_for_llm.py script.

Data Ingestion: Raw PDF documents are loaded from a designated /data directory using LangChain's DirectoryLoader and PyPDFLoader.

Text Segmentation: To ensure the context remains manageable for the LLM, the documents are split into chunks using the RecursiveCharacterTextSplitter. A chunk size of 500 characters and a chunk overlap of 50 characters are applied to maintain semantic continuity between adjacent segments.

Vector Transformation and Storage: The text chunks are converted into embeddings using the HuggingFace all-MiniLM-L6-v2 model. These embeddings are then indexed in a local FAISS vector store, which is saved to vectorstore/db_faiss for subsequent retrieval.

3. Phase 2: Retrieval and LLM Integration

This phase establishes the bridge between the stored vectors and the generative model, implemented in connect_memory_with_llm.py.

Vector Store Retrieval: The pre-built FAISS index is loaded into memory. When a query is received, the system retrieves the top-3 most relevant document chunks based on L2 similarity search.

Prompt Orchestration: A specialized retrieval-qa-chat prompt is pulled from the LangChain hub. This prompt instructs the LLM to use only the provided context for answering, effectively constraining the model's generation to the medical knowledge base and reducing hallucinations.

Inference Configuration: The llama-3.1-8b-instant model is initialized with a temperature of 0.5 and a maximum token limit of 512 to ensure a balance between precision and natural language flow.

4. Phase 3: User Interface Deployment

The final implementation layer is the interactive web dashboard created in medibot.py.

State Management: Streamlit's st.session_state is used to maintain the chat history throughout the session.

Resource Caching: To optimize performance, the FAISS vector store and embedding models are wrapped in @st.cache_resource. This ensures that these heavy resources are only loaded once, significantly reducing the system's startup time and memory footprint.

RAG Chain Execution: The UI integrates the retrieval and generation components into a rag_chain. When a user submits a prompt, the system displays the resulting answer along with the source documents to provide full transparency and traceability for the provided medical information.

Summary of System Parameters

Parameter	Configuration
Embeddings Model	sentence-transformers/all-MiniLM-L6-v2
Chunk Strategy	500 characters / 50 overlap
Search Method	FAISS L2 Similarity Search
Top-K Retrieval	k=3
Generative LLM	Llama 3.1 8B (via Groq)
Interface	Streamlit

EXPECTED OUTCOMES

Reduction in Hallucinations: By constraining the Llama-3.1 model to answer solely based on the retrieved FAISS context, the system avoids inventing medical facts. If the answer is not in the documents, the model is instructed to state "I don't know."

Low Latency Response: The use of Groq's LPU technology allows for token generation speeds significantly faster than traditional GPU-based cloud APIs, making the chatbot feel instantaneous.

Source Traceability: The system is capable of returning the specific source documents (metadata and page content) used to generate an answer, fostering user trust in the medical advice provided.

Offline Privacy: By calculating embeddings locally with all-MiniLM-L6-v2 and storing data in a local FAISS index, sensitive medical documents never leave the local environment during the indexing phase.

FUTURE SCOPE

Multi-Modal Capabilities: Future iterations could incorporate image recognition to analyze medical scans or prescriptions, similar to the multi-modal approaches discussed in recent literature.

Authentication: Adding user login via Streamlit to secure access to the bot.

Hybrid Search: Implementing a hybrid search (keyword + semantic) to improve retrieval accuracy for specific medical drug names or terminology.

Federated Learning: To further protect patient privacy, future models could utilize federated learning, where the AI learns from decentralized data without removing sensitive records from local devices.

CONCLUSION

The development of MediBot represents a successful implementation of Retrieval-Augmented Generation (RAG) to solve one of the most pressing issues in medical AI: the lack of factual grounding in Large Language Models. By decoupling the reasoning engine from the knowledge base, this research has demonstrated that it is possible to provide context-aware, evidence-based answers that are directly traced to validated medical literature. Technical analysis of the system confirms that the modular three-phase architecture comprising PDF ingestion, FAISS vector indexing, and Groq-accelerated inference offers a scalable solution for clinical decision support. Specifically, the use of sentence-transformers/all-MiniLM-L6-v2 for embeddings paired with Llama 3.1 ensures that the system maintains high semantic accuracy while delivering sub-second response times. This low-latency performance is crucial for real-time healthcare environments where delayed information can impact patient outcomes.

Furthermore, the introduction of a Streamlit-based interface with source document traceability addresses the critical "black-box" problem of traditional AI. By allowing clinicians and students to verify the model's claims against raw document snippets, MediBot fosters a culture of transparency and trust that is essential for the adoption of AI in medicine. While generic LLMs continue to struggle with "hallucinations," the constrained retrieval pipeline implemented here ensures that the model remains a reliable assistant rather than a primary decision-maker. Future iterations of this work will explore the integration of sentiment analysis to enhance empathetic communication, as well as the implementation of unit testing for RAG applications to further harden the system against edge-case errors. Ultimately, MediBot serves as a functional blueprint for the next generation of medical copilots, proving that with the right architectural constraints, AI can be a safe and indispensable tool in modern clinical practice.

REFERENCES

[1] Ahamed, S., et al. (2026). SuSastho.AI: A multimodal medical copilot for adolescents using evidence-based medicine and large language models. *Informatics in Medicine Unlocked*, 60, 101720.

[2] Steybe, D., et al. (2025). Evaluation of a context-aware chatbot using retrieval-augmented generation for answering clinical questions on medication-related osteonecrosis of the jaw. *Journal of Cranio-Maxillo-Facial Surgery*, 53, 355-360.

[3] Younesi, A., et al. (2026). HealthCare 5.0: An Industry 5.0 perspective for next-generation medical systems with synergistic integration of IoT, AI, and 6G. *Internet of Things*, 35, 101815.

[4] Nayinzira, J. P., & Adda, M. (2024). SentimentCareBot: Retrieval-Augmented Generation Chatbot for Mental Health Support with Sentiment Analysis. *Procedia Computer Science*, 251, 334-341.

[5] Martinez Amodia, R., et al. (2025). RAGBOT CLI: a Python library for running and evaluating retrieval-augmented generation chatbots. *SoftwareX*, 32, 102458.

[6] Hovland R, Bremer S, Frigaard C, et al. Effect of a pharmacist-led intervention on adherence among patients with a first-time prescription for a cardiovascular medicine: a randomized controlled trial in Norwegian pharmacies. Int J Pharm Pract. 2020;28(4):337–345.

[7] Street Jr RL, Makoul G, Arora NK, Epstein RM. How does communication heal? Pathways linking clinician-patient communication to health outcomes. Patient Educ Couns. 2009;74(3):295–301.

[8] Kaae S, Rossing C, Husted GR, Fosgerau CF. How patient-centredness takes place in pharmacy encounters: a critical common-sense interpretation of videorecorded meetings. Int J Clin Pharmacol. 2023;45(1):146–153.

[9] Greenhill N, Anderson C, Avery A, Pilnick A. Analysis of pharmacist-patient communication using the Calgary-Cambridge guide. Patient Educ Couns. 2011;83(3): 423–431.

[10] Wallman A, Vaudan C, Sporrong SK. Communications training in pharmacy education, 1995-2010. Am J Pharm Educ. 2013;77(2):36. https://doi.org/10.5688/ ajpe77236.

[11] Kurtz S, Draper J, Silverman J. Teaching and Learning Communication Skills in Medicine. 2nd ed. Radcliffe Publishing; 2005.

[12] Svensberg K, Sporrong SK, Lupattelli A, Olsson E, Wallman A, Bjornsdottir ¨ I. Nordic pharmacy students' opinions of their patient communication skills training. Am J Pharm Educ. 2018;82(2):6208.

[13] Svensberg K, Bjornsdottir ¨ I, Wallman A, Sporrong SK. Nordic pharmacy schools' experience in communication skills training. Am J Pharm Educ. 2017;81(9):6005.

[14] Abolfazl Younesi et al. HealthCare 5.0: An industry 5.0 perspective for next-generation medical systems with synergistic integration of IoT, AI, and 6G
Internet of Things Volume 35, January 2026, 101815